Principal Investigator(s) Name(s): Prithviraj Banerjee
PI Institution: University of Illinois at Urbana-Champaign
PI Phone Number: (217) 333-6564
PI E-mail Address: banerjee@crhc.uiuc.edu
Grant or Contract Title: A Novel System Level Approach to Fault Tolerance in Distributed Memory Multicomputers
Grant or Contract Number: N00014-91J-1096
Modification: A00001
Funding Level: $431,000 for 3 years
Grant Period: 1 Oct 90 - Sep. 30 1993 (extension to 31 July 94)

A NOVEL SYSTEM LEVEL APPROACH TO FAULT TOLERANCE IN DISTRIBUTED MEMORY MULTICOMPUTERS

FINAL REPORT SUBMITTED TO

Department of the Navy
Office of Naval Research
Federal Building, Rm. 286
536 S. Clark St.
Chicago, IL-60605-1588

AND TO

ONR Program Director: Dr. Gary Koob
Computer Science Division, Code 1133SS
Office of Naval Research
800 N. Quincy St.
Arlington, VA-22217-5000

REPORT PREPARED BY

Investigator: Prithviraj Banerjee

Center for Reliable and High-Performance Computing
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1308 W. Main St.
Urbana, Illinois 61801
(217) 333-6564

September 1994

Principal Investigator(s) Name(s): Prithviraj Banerjee
PI Institution: University of Illinois at Urbana-Champaign
PI Phone Number: (217) 333-6564
PI E-mail Address: banerjee@crhc.uiuc.edu
Grant or Contract Title: A Novel System Level Approach to Fault Tolerance in Distributed Memory Multicomputers
Grant or Contract Number: N00014-91J-1096
Funding Level: $431,000 for 3 years
Reporting Period: 1 Oct 90 - 31 July 94

1. DETAILED SUMMARY OF PROGRESS

The objective of this research was to develop new, cost-effective techniques for fault tolerance in multicomputer architectures. The requirements for high performance and fault tolerance are seemingly contradictory: parallel architectures and algorithms developed for high performance attempt to achieve maximum utilization of each of the processors, while fault tolerance requires redundant computations and checks to ensure that the results of the computations are correct. The result is that conventional fault tolerance techniques are very expensive when applied to highly parallel multicomputer architectures. Our unique approach to achieve fault tolerance in multicomputer parallel architectures is to use an algorithm-based fault tolerance (ABFT) technique which is an on-line system-level method for detection of faults followed by a system level approach to reconfiguration and recovery of a parallel processor system.

In the past three and half years, this research pursued four major topics: (1) An evaluation of algorithm-based fault tolerance on general purpose multiprocessors in the presence of round-off errors. (2) Development of novel algorithm-based fault tolerance techniques for solutions of partial differential equations. (3) Development of compiler-assisted automated synthesis of algorithm-based fault detection techniques; (4) Investigation of system-level reconfiguration and recovery techniques; In the following we give some details of the research results obtained in the previous year.

1.1. Evaluation of algorithm-based fault tolerance in the presence of round-off errors

Algorithm-based techniques are based on checking for the preservation of certain properties possessed by global data following a set of computations. This often involves the introduction of a check variable which is updated in such a manner that, in the absence of roundoff errors, it equals the value of some function which involves all the data elements participating in the algorithm. Usually, the function chosen is a sum over all data elements, known as a checksum, since this is both easy to compute and leads to easy determination of the rules for updating the check variables. However, the fact that roundoff errors accumulate in different ways in the updates involving the check variables and the computations involving data elements make it highly unlikely that the equality is preserved exactly for an implementation of the algorithm on a real computer. Thus, the check step involves verifying the preservation of the equality to within a tolerance value. So far, two approaches have been taken for the determination of the tolerance. One is an experimental method which requires the data sets involved to be of fixed size and range to be effective. Another more recent approach has suggested extracting the mantissas of the floating point quantities involved and applying an exact integer checksum test to products involving mantissas prior to floating point summation. However, errors in the floating point summations cannot be easily checked. In this research, we proposed a method for determination of the tolerance based on error analysis techniques. In the interest of rapid derivation of the tolerance expressions, some simplifications are introduced in the analysis process which, however, do not lessen the effectiveness of the tolerance expression derived. We present results on three numerical algorithms which show the effectiveness of our approach for data sets of varying sizes and data ranges. We have applied this technique to three parallel applications on an Intel iPSC hypercube multiprocessor, namely, Matrix multiplication, Gaussian Elimination and QR factorization. Error coverages of about 90-100% were observed over a wide range of data sizes and data ranges.

1.2. Investigation of Novel Schemes of Algorithm-based fault tolerance

Algorithm-based schemes have been proposed for a wide variety of numerical applications by us and other researchers in the past. Applications that were developed in the past include matrix multiplication, Fast Fourier Transform, Gaussian elimination, QR factorization, filtering, etc. Those schemes were all based on either the

checksum encoding or the sum-of-squares encoding.

However, for a particular class of numerical applications, namely those involving the iterative solution of linear systems, there exist almost no fault-tolerant algorithms in the literature. In this research, we describe a fault-tolerant version of a parallel algorithm for iteratively solving the Laplace equation over a grid. The fault-tolerant algorithm is based on the popular successive overrelaxation scheme with red-black ordering. We use the Laplace equation merely as an illustration; fault-tolerant versions of other iterative schemes for solution of linear systems arising from discretizations of other partial differential equations may be similarly derived.

We also present a new way of dealing with the roundoff errors which complicate the check phase of algorithm-based schemes. Our approach is based on error analysis incorporating some simplifications and gives high fault coverage and no false alarms for a large variety of data sets, as shown by our results.

The timing overheads of our fault-tolerant algorithm over the basic SOR algorithm involving no fault tolerance decrease with increasing problem dimension and become negligible for large data sizes.

1.3. Compiler Assisted Synthesis of ABFT techniques

All the algorithm-based fault detection techniques proposed in the past have been application and algorithm specific; they had been designed by the user by exploiting some particular features of the algorithms. If one wishes to apply these techniques to a large set of useful parallel applications on real parallel machines, each of these parallel applications will have to be rewritten with ABFT checks. Such a procedure would be clearly quite tedious for the user and therefore not very useful. In this research, we are therefore exploring the possibility of automating the process of designing the ABFT checks on existing application programs.

We have proposed a theoretical basis for synthesizing algorithm-based checking techniques for general applications at the compiler level. The basic approach is to identify linear transformations in Fortran DO loops, restructuring program statements to convert non-linear transformations to linear ones, and inserting system-level checks based on this property. In previous years, we had developed the framework of such a compiler.

We have implemented a source-to-source restructuring compiler, CRAFT (CompileR assisted Algorithm-based Fault Tolerance), for the synthesis of low-cost system-level checks for general numerical Fortran programs, based on the above approach. We have used Parafrase-2, an existing source-to-source vectorizing package, to aid us in the CRAFT project.

The LINTEST pass of our compiler performs automatic detection of linear statements in a given program followed by linearization of other statements. LINTEST is executed in two phases or sub-passes. The first of these is the GETLIN phase, which goes through the entire program and identifies symbolically linear statements. The second phase uses the above information and calls one of the following two procedures, depending on the nature of the statement currently being processed: (1) the PROGLIN procedure; (2) the MAKELIN procedure. The GETLIN phase performs step-by-step detection of symbolic linearity for all assignment statements (inside loops). The ADDCHECK pass of the CRAFT compiler operates on the linearized version of the original program and generates suitable check code for providing on-line detection of errors that may occur during actual execution of the program.

We have applied some of the compiler assisted synthesis of algorithm based checks to some real scientific FORTRAN programs. Some sample routines from the LINPACK library, (namely, DGEFA and DGESL) and EISPACK library (namely TRED2 and TQL2), and two programs from the Perfect Club Benchmark Suite (namely TRFD and MDG and their subroutines), were considered. For each of those applications, we have obtained detailed experimental results.

The main thrust of our compiler assisted approach has been to producing algorithm-based checks, hence our compiler only inserts checks to an already existing parallel program. To make the process more powerful, one can use such a compiler as a backend to a parallelizing compiler. The status of parallelizing compilers for distributed memory machines for which our algorithm based checking methods are applicable is quite premature. Unfortunately no such mature compiler exists that we could use. Hence an interesting research that we started this year was to adapt some of the recent advances in distributed memory compilers and implement those ideas in our PARADIGM compiler.

The PARADIGM compiler project at Illinois provides an automated means to convert sequential programs for execution on distributed memory multicomputers. This is accomplished in two steps. The first step involves the automatic parallelization of the sequential program into a shared memory parallel program using traditional compiler dependence analysis. The second step involves the data partitioning, computation partition, and communication generation steps for the parallel program for efficient execution on distributed memory machines. The compiler is targeted to structured parallel numerical applications written in FORTRAN 77 that operate on regular data structures such as matrices. The sequential FORTRAN programs for regular applications are automatically parallelized using a parallelizing compiler (we use Parafrase-2), and the PARADIGM compiler performs several compiler transformations on the program, and then generates efficient message passing FORTRAN code. The basic framework of the compiler is similar to the FORTRAN D compiler from Rice University. Numerous compiler optimizations such as loop bounds reduction, message vectorization, message chaining, message aggregation and pipelining, are automatically performed by the PARADIGM compiler.

In addition, the PARADIGM compiler is unique in its ability (1) to perform automated data distribution for regular data structures, something that has to be conventionally specified as a compiler directive in FORTRAN D and High Performance FORTRAN; (2) to generate high-level communication primitives, such as EXPRESS library calls; (3) Simultaneous exploitation of functional and data parallelism; (4) Generating multithreaded message-driven code (as opposed to conventional SPMD message send-receive code) to hide the long latencies of communication, by overlapping computation with communication. We have already implemented such a compiler which generates code for machines such as the Intel iPSC/860, the Intel Paragon, and the Connection Machine CM-5.

1.4. System Level Reconfiguration and Recovery Techniques

Once a faulty processor has been detected using our algorithm based scheme, one needs to investigate schemes for reconfiguring the system around the faulty processors. We have proposed two hardware approaches and a software approach for reconfiguring hypercube and mesh based multiprocessors.

In the first hardware scheme for reconfiguration, we assume that spare processors can be attached to specific processors of the hypercube or mesh. In this approach, we designate two types of nodes (P-nodes and S-nodes) in the hypercube or mesh. P-nodes correspond to conventional processing nodes in distributed memory systems. S-nodes correspond to nodes that have a spare processor and memory attached to the node. The message routing logic is shared between the two processing units. If we assign one spare processor to a set of active processors, then the hardware overhead can be minimized. We have developed appropriate embedding algorithms for hypercube and mesh topologies.

The second hardware approach places spare processors along specific links in the hypercube or mesh. Hence the node hardware is not changed at all. We have investigated appropriate embeddings of cost effective ways to place spare processors in selected links of a hypercube and mesh. We have again formally developed reconfiguration algorithms for this framework.

Both the hardware schemes involve mapping of logical links of a virtual machine onto a set of physical links in the final reconfigured machine and hence suffer some performance degradation. We have analyzed the performance degradation theoretically and experimentally. Our theoretical studies have led to bounds on the dilation of the logical links in both the schemes. We have performed simulation studies to verify these bounds. The simulations have been performed on several large parallel applications running on an Intel iPSC/2 hypercube.

A third scheme for reconfiguration that we have proposed in this research has been on a software reconfiguration strategy for hypercube multicomputer architectures under multiple faults. The advantage of this reconfiguration strategy over previous reconfiguration schemes is that it requires no redundant hardware, but supports reconfiguration through graceful degradation. It is based on the notion of using multiple virtual processors on a single physical processor and using these virtual processors for workload redistribution under faults. We have actually implemented a software reconfiguration scheme based on the notion of virtual processors. We have performed a case study of the performance degradation of the reconfiguration scheme on a commercially available Intel iPSC/2 hypercube multicomputer for several actual parallel applications.

Finally, we have been investigated a dynamic software reconfiguration and recovery scheme using the object-oriented actor model of computation. The actor programming model involves a message driven style of

execution, where each task represents an amount of work that is carried on until completion. At the end of the execution of the current actor, messages are sent to other actors in an asynchronous manner. Each processors sits in a pick-next-actor loop executing messages.

We have used the above parallel programming environment to develop a system level reconfiguration scheme. For every actor task in the original system, we create a shadow actor task which gets automatically placed on a different processor. Any message communication between original actor tasks are also sent to the shadow actors. The shadow actors are passive replicas of the original tasks, and do not actually execute anytime under normal conditions. Under a fault, all actors currently executing on the faulty processor are killed, and their shadows on different processors are initiated to continue the work. We have implemented such a system using the CHARM parallel programming systems, and have evaluated the overheads of our scheme on a set of complex parallel applications.

Principal Investigator(s) Name(s): Prithviraj Banerjee
PI Institution: University of Illinois at Urbana-Champaign
PI Phone Number: (217) 333-6564
PI E-mail Address: banerjee@crhc.uiuc.edu
Grant or Contract Title: A Novel System Level Approach to Fault Tolerance in Distributed Memory Multicomputers
Grant or Contract Number: N00014-91J-1096
Funding Level: $431,000 for 3 years
Reporting Period: 1 Oct 90 - 31 July 94

2. PRODUCTIVITY MEASURES

Refereed journal papers submitted but not yet published: 4

Refereed journal papers published: 4

Refereed conference papers published: 18

Unrefereed reports and articles: 0

Books or parts thereof submitted but not yet published: 0

Books or parts thereof published: 2

Patents filed but not yet granted: 0

Patents granted (include software copyrights): 0

Invited presentations: 5

Contributed presentations: 18

Honors received: 20

Description of Honors:

- Awarded the *1994 Outstanding Paper Award* from the International Conference on Parallel Processing, St. Charles, IL, August 1994.
- Awarded the *1992 University Scholar Award* from the University of Illinois.
- Recipient of the *1992 Senior Xerox Award for Faculty Research*, University of Illinois.
- Elected to *Senior Member*, IEEE, 1990.
- Associate Editor, IEEE Transactions on VLSI Systems, 1993, 1994
- Associate Editor, Journal of Parallel and Distributed Computing, 1993, 1994
- Associate Editor, Journal of Circuits, Systems and Computers, 1993
- Technical Program Area Chair, {\m Int. Symp. Circuits and Systems (ISCAS-93),} (Chicago, Illinois, May 1993).

• Program Committee Member, {\m 9th Int. Parallel Processing Symp. (IPPS-95),} Santa Barbara, CA, Apr. 1995.

• Program Committee Member, {\m 7th Int. Conf. on VLSI Design} (New Delhi, INDIA, Jan. 1995).

• Organizing and Program Committee Member, {\m 21st Int. Symp. on Computer Architecture,} (Chicago, IL, May 1994).

• Program Committee Member, {\m 8th Int. Parallel Processing Symp. (IPPS-94),} Cancun, Mexico, Apr. 1994.

• Organizing and Program Committee Member, {\m 6th Int. Conf. on VLSI Design} (Calcutta, INDIA, Jan. 1994).

• Program Committee Member, {\m 23rd Int. Symp. Fault Tolerant Computing} (Tolousse, FRANCE, June 1993).

• Program Committee Member, {\m 5rd Int. Conf. on VLSI Design} (Bombay, INDIA, Jan. 1993).

• Program Committee Member, {\m 19th Int. Symp. on Computer Architecture,} (Queensland, Australia, May 1992).

• Program Committee Member, {\m Int. Workshop on Fault Tolerance in Parallel and Distributed Systems,} (Amherst, MA, Jul. 1992).

• Program Committee Member, {\m 18th Int. Symp. on Computer Architecture} (Toronto, CANADA, May 1991).

• Program Committee Member, {\m 5th Int. Parallel Processing Symp.} (Orange County, California, Mar. 1991).

• Program Committee Member, {\m 3rd Int. Symp. on VLSI Design} (New Delhi, INDIA, Jan. 1991).

Prizes/Awards Received: 3

Description of Award:

• Received the 1992-93 University Scholar Award from University of Illinois in recognition of outstanding research and teaching

Promotions Obtained: 2

Description of Promotion: Promoted to Full Professor

Promoted to Director of Computational Science and Engineering Program, UIUC

Graduate students supported >= 25% of full time: 6

Post docs supported: 0

Minorities supported include Black Hispanic American Indians and other native Americans such as

Aleuts, Pacific Islanders, etc.; do not include Asians or Asian-Americans): 0

Principal Investigator(s) Name(s): Prithviraj Banerjee
PI Institution: University of Illinois at Urbana-Champaign
PI Phone Number: (217) 333-6564
PI E-mail Address: banerjee@crhc.uiuc.edu
Grant or Contract Title: A Novel System Level Approach to Fault Tolerance in Distributed Memory Multicomputers
Grant or Contract Number: N00014-91J-1096
Funding Level: $431,000 for 3 years
Reporting Period: 1 Oct 90 - 31 July 94

3. LIST OF PUBLICATIONS

BOOK CHAPTERS

[1] D. Pradhan and P. Banerjee, "Fault Tolerant Multiprocessor Design Issues," chapter in *Fault Tolerant System Design*, Editor: D. Pradhan, Prentice-Hall, Englewoods Cliffs, NJ, 1994.

[2] P. Banerjee, A. Roy-Chowdhury and V. Balasubramanian, "Compiler-Assisted Synthesis of Algorithm-Based Fault Tolerance," chapter in *Foundations of Ultradependable Computing, Volume III: System Implementations*, Editors: G. Koob and C. Lau, Kluwer Academic Publishers, 1994.

ARTICLES IN JOURNALS

[3] M. Gupta and P. Banerjee,"Demonstration of Automated Data Partitioning Techniques in Parallelizing Compilers for Distributed Memory Multiprocessors," *IEEE Trans. Parallel and Distributed Systems*, March, 1992, Vol. 3, no. 2, pp. 179-193.

[4] P. Banerjee and M. Peercy, "Design and Evaluation of Hardware Strategies for Reconfiguring Hypercubes and Meshes Under Faults," *IEEE Trans. Computers*, Volume 43, Number 7, July 1994, pp. 841-848.

[5] M. Peercy and P. Banerjee,"Fault Tolerant VLSI Systems," *Proceedings of the IEEE (Special Issue on VLSI Reliability)*, (invited paper), May 1993, Volume 81, number 5, pp. 745-758.

[6] M. Gupta and P. Banerjee, "Compile-time Estimation of Communication Costs in Programs," *Jour. Programming Languages*, to appear.

[7] A. Roy-Chowdhury and P. Banerjee, "A New Error Analysis Based Method for Tolerance Computation for Algorithm-Based Checks," *IEEE Trans. Computers*, submitted.

[8] A. Roy Chowdhury and P. Banerjee, "A Fault Tolerant Parallel Algorithm for Iterative Solution of the Laplace Equation," *IEEE Trans. Parallel and Distr. Systems*, submitted.

[9] A. Roy Chowdhury and P. Banerjee, "Algorithm-based Fault Location and Recovery for Matrix Computations," *IEEE Trans. Computers*, submitted.

[10] M. Peercy and P. Banerjee, "A Software Strategy for Reconfiguring Hypercubes Under Faults," *Jour. Parallel Dist. Computing*, submitted.

CONFERENCE PROCEEDINGS

[11] M. Gupta and P. Banerjee, "Automated Data Partitioning on Distributed Memory Multiprocessors," *Proc. 6th Distributed Memory Multicomputers Conference (DMMC6)*, Portland, OR, May 1991.

[12] V. Balasubramanian and P. Banerjee, "CRAFT: A Compiler for Synthesizing ALgorithm-Based Fault Tolerance in Hypercube Multiprocessors," *Proc. Int. Conf. Parallel Processing (ICPP-91)*, St. Charles, IL, Aug. 1991.

[13] M. Gupta and P. Banerjee, "Compile-Time Estimation of Communication Costs in Multicomputers,," *Proc. Int. Parallel Proc. Symp.* Beverly Hills, CA, Mar. 1992.

[14] M. Gupta and P. Banerjee, "A Methodology for Synthesis of High-Level Communication in Distributed Memory Multicomputers," *Proc. 6th ACM Int. Conf. Supercomputing (ICS-92)*, Jul. 1992, Washington, DC.

[15] M. Peercy and P. Banerjee, "Design and Analysis of Software Reconfiguration Strategies of Hypercube Multiprocessors Under Multiple Faults," *Proc. 22nd Fault Tolerant Computing Symp.*, Jul. 1992, Boston, MA.

[16] A. Roy Chowdhury and P. Banerjee,"Tolerance Determination for Algorithm-based Checks Using Simplified Error Analysis," *Proc. Fault-Tolerant Computing Symposium (FTCS-93)*, Toulouse, FRANCE, Jun. 1993.

[17] M. Gupta and P. Banerjee,"PARADIGM: A Compiler for Automated Data Partitioning", *Proc. Int. Conf. Supercomputing (ICS-93)*, Tokyo, JAPAN, Jul. 1993.

[18] S. Ramaswamy and P. Banerjee,"Processor Allocation and Scheduling of Macro Dataflow Graphs on Distributed Memory Multicomputers by the PARADIGM Compiler", *Proc. Int. Conf. Parallel Processing (ICPP-93)*, St. Charles, IL, Aug. 1993.

[19] J. Chandy and P. Banerjee,"Reliability Evaluation of Disk Array Architectures," *Proc. Int. Conf. Parallel Processing (ICPP-93)*, St. Charles, IL, Aug. 1993.

[20] A. Roy Chowdhury and P. Banerjee,"A Fault Tolerant Parallel Algorithm for Iterative Solution of the Laplace Equation," *Proc. Int. Conf. Parallel Processing (ICPP-93)*, St. Charles, IL, Aug. 1993.

[21] E. Su, D. Palermo, and P. Banerjee,"Automatic Parallelization of Regular Computations for Distributed Memory Multicomputers in the PARADIGM Compiler," *Proc. Int. Conf. Parallel Processing (ICPP-93)*, St. Charles, IL, Aug. 1993.

[22] J. Holm, A. Lain, and P. Banerjee, "Compiling Scientific Computations into Multithreaded and Message Driven Computation," *Proc. Scalable High Performance* Computing Conf., Knoxville, TN, May 1994.

[23] A. Roy Chowdhury and P. Banerjee, "Algorithm-based Fault Location and Recovery for Matrix Computations," *Proc. Fault Tolerant Computing Symp.*, Austin, TX, July 1994.

[24] T. Karnik, S. Ramaswamy, S. M. Kang, and P. Banerjee, "Application of Algorithm Based Fault Tolerance Techniques to High Level Synthesis of Signal Flow Graphs," *Proc. SPIE Int. Symp. Advanced Signal Processing Algorithms Architectures* Implementations V, San Diego, CA, July 1994, pp. 760-776.

[25] A. Lain and P. Banerjee, "Techniques to Overlap Computation and Communication in Irregular Iterative Applications," *Proc. Int. Conf. Supercomputing*, Manchester, England, July 1994.

[26] E. Su, D Palermo, and P. Banerjee, "Processor Tagged Descriptors: A Data Structure for Compiling for Distributed Memory Multicomputers," *Proc. Conf. Parallel Architectures Compilation Techniques,* Montreal, Canada, Aug. 1994.

[27] D. Palermo, E. Su, and P. Banerjee, "Communication Optimizations for Distributed Memory Multicomputers used in the PARADIGM Compiler," *Proc. Int. Conf. Parallel Processing,* St. Charles, IL, Aug. 1994. THIS PAPER RECEIVED THE OUTSTANDING PAPER AWARD AT ICPP-94.

[28] P. Banerjee, J. Chandy, M. Gupta, J. G. Holm, A. Lain, D. J. Palermo, S. Ramaswamy and E. Su, "The PARADIGM Compiler for Distributed Memory Message-Passing Multicomputers," *Proc. First Int. Workshop on Parallel Processing,* Bangalore, INDIA, Dec. 1994.

Principal Investigator(s) Name(s): Prithviraj Banerjee
PI Institution: University of Illinois at Urbana-Champaign
PI Phone Number: (217) 333-6564
PI E-mail Address: banerjee@crhc.uiuc.edu
Grant or Contract Title: A Novel System Level Approach to Fault Tolerance in Distributed Memory Multicomputers
Grant or Contract Number: N00014-91J-1096
Funding Level: $431,000 for 3 years
Reporting Period: 1 Oct 90 - 31 July 94

4. TRANSITIONS AND INTERACTIONS

The principal investigator's work in algorithm-based fault tolerance has been picked up by researchers in General Electric (Dr. A. Chatterjee and Dr. M. d'Abreau). GE researchers have applied the fault tolerance research to their signal processing systems.

The principal investigator's work on algorithm based fault tolerance has been followed up by researchers in other universities, namely, Stanford (Prof. E. McCluskey), Princeton (Prof. N. K. Jha), SUNY Buffalo (Prof. S. S. Ravi), and University of Texas at Austin (Prof. J. Abraham)

The work on the PARADIGM compiler has been picked up by several universities and companies, namely, Rice Univ. (Prof. Kennedy), Syracuse Univ. (Prof. Fox and Prof. Choudhary), Ohio State (Prof. Sadayappan), Motorola (Dr. Natarajan). The work on the CRAFT back-end to the PARADIGM compiler has been picked up by the Univ. of Pittsburg (Prof. Melhem and Prof. Gupta).

Principal Investigator(s) Name(s): Prithviraj Banerjee
PI Institution: University of Illinois at Urbana-Champaign
PI Phone Number: (217) 333-6564
PI E-mail Address: banerjee@crhc.uiuc.edu
Grant or Contract Title: A Novel System Level Approach to Fault Tolerance in Distributed Memory Multicomputers
Grant or Contract Number: N00014-91J-1096
Funding Level: $431,000 for 3 years
Reporting Period: 1 Oct 90 - 31 July 94

5. SOFTWARE AND HARDWARE PROTOTYPES

Several software packages are being developed from this research over the past three years.

1. The CRAFT compiler has being developed which uses the PARAFRASE parallelizing compiler as a front end to parse sequential FORTRAN programs, and analyze the statements for linearity, and apply linearity based checks to the programs running on individual node processors of a distributed memory multiprocessor.

2. A parallelizing compiler for distributed memory machines called PARADIGM has been developed. The compiler performs automated data partitioning, exploits function and data parallelism, and optimizes communication in many ways.

3. An experimental software environment is being developed for analyzing actual fault and error coverage figures for real parallel applications using error injection in various types of functional elements such as floating point units, integer units, memories, communication paths, control units etc. All this is done in the presence of finite precision errors. The testbed on which this software runs is the Intel iPSC/2 hypercube.

4. Several software means of reconfiguration has been developed on an Intel iPSC/2 hypercube. One uses a static reconfiguration strategy. The second uses a dynamic strategy.